1. Honor code (2 pts)

Print your name to acknowledge the Duke Community Standard

2. (18 pts) Complete or answer the following.

\[
\begin{align*}
L_1 &= \{ba, cb\}, \Sigma = \{a, b, c\} \\
L_2 &= \{aba, cb, cc\}, \Sigma = \{a, b, c\} \\
L_3 &= b(ba)^*b, \Sigma = \{a, b\} \\
L_4 &= (a + b)b^*(a + b), \Sigma = \{a, b\} \\
L_5 &= \{w \in \Sigma^* | n_b(w) > n_a(w) + 1\}, \Sigma = \{a, b\}
\end{align*}
\]

Type your answers into the boxes. For empty set use \{\}. For empty string use &.

(a) \(| L_1 \cup L_2 | = \text{Answer is 4, the set is } \{ba, cb, aba, cc\}\)

(b) \(L_1 \circ \{ba\} = \{baba, cbba\}\)

(c) \(L_1 \cap L_2 = \{cb\}\)

(d) \(L_3 \cap L_4 = \{bb\}\)

(e) \(L_3 \cap L_5 = L_3\)

(f) \(L_1 - L_2 = \{ba\}\)

(g) \(| L_1 \circ L_2 \circ L_2 | = 18\)

(h) \(L_1 \times L_1 = \{(ba, ba), (ba, cb), (cb, ba), (cb, cb)\}\)

(i) \(| 2^{L_2} | = 8\)
3. (16 pts) Answer TRUE or FALSE to each of the statements below.

(a) Any NFA $M_1$ can be converted into a DFA $M_2$ such that $M_2$ has at most twice as many states as $M_1$. (TRUE or FALSE?)
   
   Answer: False

(b) Suppose DFA $M$ is such that $L(M)$ is an infinite language. Then $M$ could have an infinite number of states. (TRUE or FALSE?)
   
   Answer: False

(c) The following grammar $G$ is a regular grammar. (TRUE or FALSE?)

   $S \rightarrow Sa | Bb | \lambda$
   $B \rightarrow Bbb | S$

   Answer: True

(d) $\lambda \in \{ba, cb\}$

   (TRUE or FALSE?)

   Answer: False

(e) $c(a + \lambda)^*b = ca^*b$

   (TRUE or FALSE?)

   Answer: True

(f) $(a + b^*a)\emptyset(ba) = aba + b^*aba$

   (TRUE or FALSE?)

   Answer: False

(g) A regular grammar with more than 10 rules must be an infinite language.

   (TRUE or FALSE?)

   Answer: False

(h) Suppose $L$ is an infinite regular language and the regular grammar for $L$ has only two variables. Then there must be at least three rules in the grammar.

   (TRUE or FALSE?)

   Answer: True
4. (8 pts) Consider the following grammar G with start variable S.

\[ S \rightarrow aaS \mid bB \mid \lambda \]
\[ B \rightarrow bbB \mid bbS \]

A) Give a derivation for the string \( aabbbb \).
\[ \text{Answer: } S \rightarrow aaS \rightarrow aabB \rightarrow aabbbB \rightarrow aabbbbbS \rightarrow aabbbbb \]

B) Give a string of length greater than 6 that is not in \( L(G) \).
\[ \text{abababab} \]

C) Give a regular expression for \( L(G) \).
Note: The regular expression \( a*b \) you would enter as: \( a \ast b \).
\[ (aa + bbb(bb)^*)^* \]
5. (8 pts) Draw a DFA for the following language. You do not have to show trap states (show the transition diagram, indicate the start state by a short arrow, and final states by double circles.)

\[ L = \{ w \in \Sigma^* \mid n_a(w) \text{ is odd and every } b \text{ must be adjacent to another } b \}, \Sigma = \{a, b\} \]

For example, \( bbabb \), \( bbbaaa \), \( aabbbbb \), and \( aaabbbbaa \) are in \( L \).

**ANSWER:**

```plaintext
4
```
6. (8 pts) Consider the following DFA.

![DFA Diagram]

a) Give one string that is accepted by this DFA that is of length 4 or greater.

**Answer Examples:** baab, aabab, bbbab
b) Consider converting this DFA into a minimal state DFA. Draw the tree that shows how states are determined to be distinguishable.

The root should be labeled: 0 1 2 3 4 5 6 7, to represent all the states from q0 to q7.

Beside or under each node, identify the terminal that splits the node into children, if that node has children, or other reason why you are splitting the node.

**ANSWER:** The first node is split between final and nonfinal nodes. The remaining nodes are split on the terminal underneath the node.

![Diagram of the tree]

c) In the algorithm to convert a DFA into a minimal state DFA, in general is it possible for a terminal to break a tree node named N into two or more child nodes, and then the same terminal be used again in one of N’s child nodes to break that node into two or more child nodes? Explain.

**Answer:** Yes, you break a node by a terminal into two or more child nodes, say you use the terminal a. Then you break another node in another part of the tree into two more child nodes. When you come back to the first node mentioned, it may now be possible to split its child node on the same terminal a since there are more nodes in the tree now the behaviour in splitting the node could now be different (go to different nodes where before those different nodes were the same node).

In part b you can see that b was used twice.
7. (8 pts) Consider the following DFA.

Note there are two arcs from $q_0$ to $q_1$, one labeled $a$ and one labeled $b$. It is shown as one arc with two labels.

a) Give an equivalent regular expression for this DFA.

**Answer:** $(b + a)(b + a(a + b))^*$

b) Give an equivalent regular grammar for this DFA.

**Answer:**

$q_0$ is start variable

$V = \{q_0, q_1\}$

$q_0 \rightarrow aq_1|bq_1$

$q_1 \rightarrow bq_1|aq_0|\lambda$
8. (8 pts) Consider the following property, Replace3rdBwithBB. If \( L \) is a regular language, then Replace3rdBwithBB(\( L \)) = strings from \( L \) that replace the third \( b \) in the string with \( bb \). If there is a string \( w \) in \( L \) that does not have at least three \( b \)'s, then that does not put a string in Replace3rdBwithBB(\( L \)).

For example, if \( abbaaba \) is in \( L \), then \( abbaabba \) is in Replace3rdBwithBB(\( L \)). (The third \( b \) is replaced with \( bb \)). If \( abababab \) is in \( L \), then \( abababbab \) is in Replace3rdBwithBB(\( L \)). If \( abbbba \) is in \( L \), then \( abbbbba \) is in Replace3rdBwithBB(\( L \)). If \( aaabb \) is in \( L \), then \( aaabb \) does not generate a string in Replace3rdBwithBB(\( L \)), because it only has two \( b \)'s.

For this problem, you only need to explain how to convert a DFA \( M \) for \( L \) into a DFA or NFA that represents the language Replace3rdBwithBB(\( L \)). That is you need to describe all the additions (copy of \( M \)?, additional states, additional arcs and their labels, etc.) and subtractions (removed states, removed arcs, etc) to create the new DFA or NFA that represents Replace3rdBwithBB(\( L \)). You DO NOT need to prove that this construction works.

**Answer:**

\[
\hat{M} = (\hat{Q}, \Sigma, \hat{\delta}, \hat{q}_0, \hat{F})
\]

\( L \) is regular, so \( L \) has a DFA \( M \). Make 3 additional copies of \( M \) called \( M' \), \( M'' \) and \( M''' \). Make a new Machine \( \hat{M} = (\hat{Q}, \Sigma, \hat{\delta}, \hat{q}_0, \hat{F}) \)
\[ \hat{Q} = Q \cup Q' \cup Q'' \cup Q''' \cup \{ p''q'' \text{ for every } \delta(p'', b) = q'' \} \]

Note: We add new states, one for every b arc in M''.

\[ \hat{F} = F''' \]

Note: All final states in M, M' and M'' are no longer final states.

\[ \hat{\delta} = \delta \cup \delta' \cup \delta'' \cup \delta''' \]

plus and minus the following:

For each \( \delta(p, b) = q \) in M. Remove that arc and add the arc \( \delta(p, b) = q' \).

For each \( \delta'(p', b) = q' \) in M'. Remove that arc and add the arc \( \delta'(p', b) = q'' \).

For each \( \delta''(p'', b) = q'' \) in M'', Add a new state called p''q''. Remove that arc and add two new arcs, \( \delta''(p'', b) = q''p'' \) and \( \delta''(q''p'', b) = p'' \).

With this construction, the first b encountered is forced into M', the second b encountered is forced into M'' and the third b encountered is forced into M''' and goes through an additional state processing bb. The only place to accept is in M''' after three b's have been processed and the third b has been replaced by bb.